Software Project from A Testing Perspective

WHITE PAPER

Author : Lakshmi.K.Kumar

Definitions of test profiles suggest ways to make the planning & management exercises of testing software projects (called testing project in this paper) more scientific. This can be achieved by introducing some quantitatively measurable parameters in each of the profiles – particularly so in the planning phase. Emphasis is on setting “measurable quality & testing goals” upfront, for the software project/product, and deriving all other attributes of the plan so as to achieve these goals.

This white paper discusses the subject of software project planning & management from a testing perspective. Three test profiles are defined for a given project, each at a different stage in the lifecycle.

Table of Contents

Introduction

This paper discusses the attributes that characterize any software project/product program, from the view point of a testing/QA organization. Three important profiles are discussed – these three profiles characterize the project/product program at three stages in the life cycle.

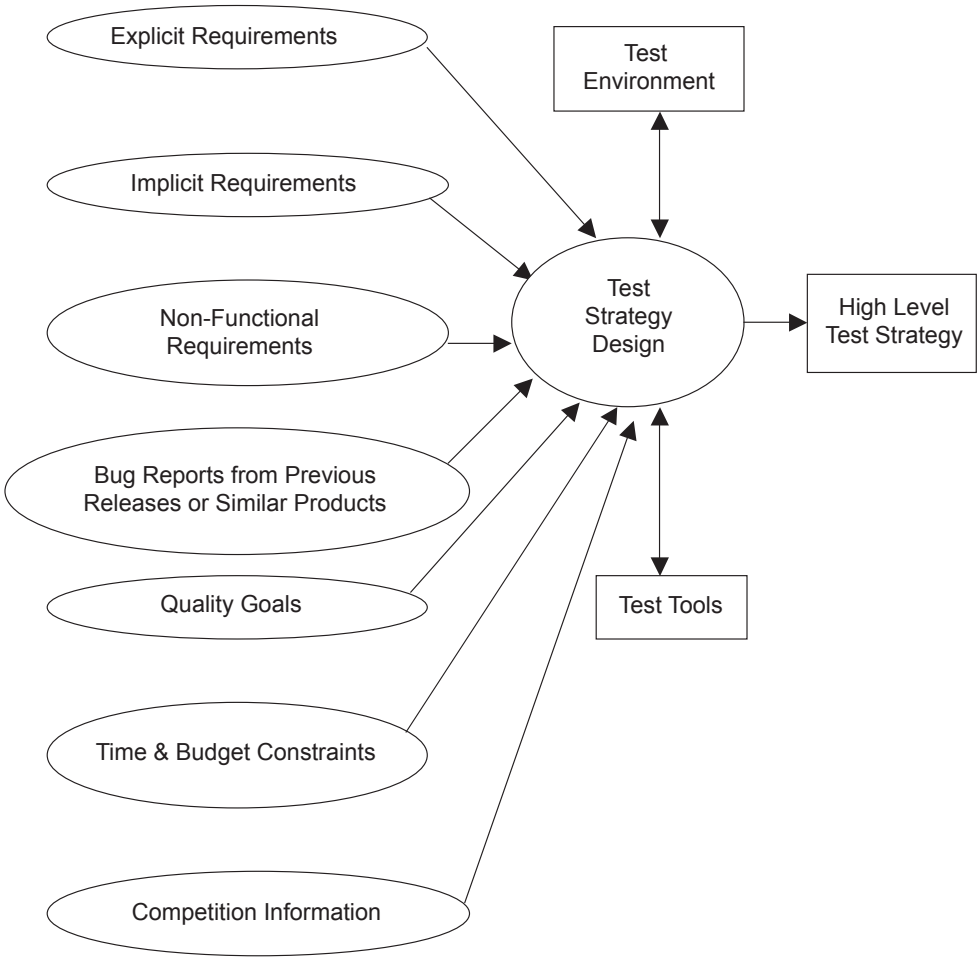
- Planned Test Profile - At project planning phase
- Execution Test Profile - During project execution
- Release Test Profile - At project closure/product release

Planned Test Profile of a Software Project

Planned Test Profile is the outcome of some high-level techno-managerial decisions made at the project planning/scoping phase. Some Important decisions that need to be made at the project planning phase are:

- Testing Budget Allocation Or Testing Cost/effort Estimation
- Usage plan for this allocated or estimated budget
- Answers for the question - How much to test? Or when to stop testing?
- Types of testing to be done
- Test design/development methodology
- Test environment & tools
- Limitations because of the above choices made
- Risks & mitigation plans

Some of the factors that will influence these decisions are shown in the picture below:



Quality v/s Time/Effort/Cost Balance for Testing Activities

The question to be answered is -
What percentage of the total project cost/ effort should be allocated for testing?

This answer depends on some of the following:

- Nature of the product/ project (mission critical or not)
- Internal quality goals set for the project/product
- Certification requirements for the product
- “Quality promise “ made to the customers of the product/ market
- “Service Level Agreements” expected to be signed for services provided using the product
- Possible “Penalty clauses” for failing to meet the quality goals/ quality promises/SLAs.

One way of deciding the test budget is to fix the “quality goal” & then estimate the cost/effort/ time/resources required for achieving this goal.

The other way is to fix the cost/effort/time/resources constraints (that is the testing budget) and then estimate the quality goals that can be achieved with these constraints. This may not be easy in the first iteration. It is possible in successive iterations, based on metric data of previous iterations.

The above factors may be narrowed down to the following headings:

- Quality goals
- Certification/conformance needs
- Testing goals
- QA load on testing

Based on the quality goals, testing goals will have to be fixed.

Quality goals

Quality Goals must be quantitatively measurable parameters. Some examples are:

- Maximum allowed residual defect density - number of defects per KLOC left behind after all testing is completed
- Maximum failure rate - Maximum possible number of failures per unit time of usage of the system (time can be calendar time or CPU time)
- MTBF - Mean Time Between Failures (number of failures per unit time)
- Reliability - probability of error free operations
- Availability - %age of the total operational time for which the system is up/functional

Certification/ conformance requirements

These are defined by the standard bodies. These specify the feature/ functionality aspects of the product. These influence test methodology decisions, pre-certification preparations, choice of certifying agent/ test-house & the cost involved.

Testing goals

The question to be answered here is -

How much to test or when to stop testing?

This is a very important decision & a hard one to be made upfront, before we start the exercise. The basis of this can be one or more of the following:

- Testing till a time/ effort/ cost budget is exhausted – Testing stops when the budget is exhausted, irrespective of the defects detected or tests executed.
- Executing a pre-defined set of test cases – This is what happens most often and the selected number or set of tests do not usually have a very logical basis. This is an area with significant improvement opportunity. This has been discussed in more detail under “test efficiency”.
- Testing till average failure rate drops below a threshold – This is followed in case of mission critical systems. No matter what the cost/ time/ effort could be, testing must continue till the “max allowed failure rate” quality goal is reached.

- Achieving a set level of coverage - Test coverage has different meanings & measurement units in the context of different kinds of testing (Ex: % KLOC, % function points, % numbered requirements) – Coverage is quantitatively measured & testing must continue till the coverage criterion is met. Test coverage has been dealt with in more detail in the appendix
- Achieving a pre-defined residual defect density - Estimating upfront, the total number of expected defects & testing till a predefined % of these defects are detected (& removed so that not more than a pre-define % of defects are left behind in the system)

Dependency of product quality on testing (QA load on testing)

Another factor to be considered is the QA load on testing. In the Software Development Life Cycle, there are other mechanisms where defect removal happens. Example: Requirement reviews, design reviews, code inspection, tool assisted design analysis & validation, validation by simulation etc.

Absence or inefficiency of these mechanisms results in poor phase containment which in turn leads to a heavy QA load on testing.

One way of determining the QA load is explained below. Let us say the “quality goal” for the product is expressed as “residual defect density (number of defects per KLOC)”.

There are techniques (Ex: COQUALMO model) for estimating the “expected defect density” & “origin (phase) of contribution of these defects”.

Defect density at the start of testing (usually system testing) can be determined by metric data of the previous phases (requirement review, design reviews, code inspection & unit test).

$$\textbf{Defect density at the start of system test} = \frac{\text{(Total number of defects estimated/ expected – total number of defects already detected/ corrected in review \& inspection)}}{\text{(Size of the software in KLOC)}}$$

$$\textbf{QA load on testing} = \frac{\text{Defect\_density at the start of system test – desired residual defect density at product release}}{\text{Estimated initial defect density}}$$

Distribution of Testing Budget

Once the total testing budget is frozen, distribution of this budget to the various aspects of testing should be decided. Not all parts/aspects/functionalities of the system need be subjected to the same degree of testing. Basis of distribution of testing budget could be one or more of the following:

- Importance and/or criticality of the module/component
- Importance and/or criticality of the functionality/ feature/business process
- Operational profile of the system
- Past test data (of previous releases) about buggy modules/ functionalities
- Quality metrics of the previous phases of SDLC of the current release (phase containment indicators)
- FMEA results (if done) of the modules/ services/ business processes

An analysis on the above lines should identify the “critical to quality” aspects of the system under test. Once the “importance/criticality” attributes of the various modules/components/ services/features/functionalities/business processes are determined, “measurable” quality & testing goals for each of these should be set.

The next step is to determine the requirements (h/w, s/w, training, tools, personnel & effort) for achieving each of these sub-goals. This in turn decides the “cost” of achieving each of these sub-goals. The total testing budget may be iteratively & judicially distributed into the sub activities.

Distribution of testing budget is a very important optimization exercise of trying to get maximum return on the investment. While making decisions on software/hardware/tool purchase, due consideration should be given to possible long term return on the investment, beyond the project in scope.

Different Types of tests that will be done

Types of tests that need to be done are based on requirements & quality goals. For example, for mission critical systems, intrusive white box testing may be recommended. The requirement could be to guarantee a maximum residual defect density or %code coverage. Or for a WEB based system, performance testing may be important.

For a COTS based system unit testing may be limited to the home-grown adaptors. For a COTS based OSS, a judicial mix of operational profile based testing & a non-operational testing may be necessary.

This decision is very specific to the project requirements and hence is left to the discretion of the working team.

Test Design/Development Methodology

This is a very important decision which can have very serious impacts. A good decision here might mean breakthrough cycle time reduction!

The essence of this is “first time automation” (not just for regression testing). The emphasis is on “automatic test generation” from FORMAL requirement/design documents or from code (Ex: UML model based test design, or automatic test generation from SDL model, automatic test generation from Message Sequence Charts,

tool generated- grammar based unit tests).This stage may involve a proof-of-concept prototyping and a ROI calculation. Complex systems benefit significantly by this approach, despite the possible initial investment cost.

The formal techniques are normally more accurate than manual methods because of their mathematical basis.

Test Environment(s) & Configurations (H/W, S/W)

This activity involves some of the following:

- Based on requirements & system architecture, identification of a test bed/ test environment
- Identification of stubs/drivers, simulators etc
- Evaluation of the environment (if one exists)
- Identifying what can & can not be tested using this environment
- Validating if staging environment mimics the real environment closely enough
- proof-of-concept prototyping (optional)

Test Tools (Type, capabilities & feature list)

Based on types of testing, test development methodology & test environment, test tools will need to be selected.

Three levels of decisions have to be made here:

- Type of the testing tool: This is based on type of testing, technology on which the product is based etc...
- ROI assessment: Here the cost of the tool & training v/s the quality & productivity improvement is assessed & hence the Return on Investment (long term use for tool should be considered) is calculated.
- Make of the testing tool: This could involve comparative study & evaluation of equivalent tools from different vendors. Some of the possible types of tools that could be of use in test & pre- test process are listed below.

- Code Inspection Tools
- Unit Test Tools
- Functional test tools
- System test tools
- Protocol test tools
- Telecom test tools (simulators/emulators/monitors
- Load/stress test tools
- Memory test tools
- Database test tools
- Web test tools
- Java test tools

Tool selection phase should involve evaluation of equivalent tools from multiple vendors, “proof-of- tool fitment” for the identified purpose & a through ROI analysis.

Execution Test Profile of a S/W Project

Execution Test Profile details attributes like testing efficiency, test automation, test management process, defect tracking, metrics collection etc...

Test Efficiency

Efficiency of test logic

This is a measure of efficiency of each test and the selected set of tests. This does not include test execution efficiency (ex: automation) or efficient use of testing resources (infrastructure & human)

It is understood that exhaustive testing is not a smart solution. Cost –effort-time constraints may force us to settle down to non-exhaustive testing. The challenge then is to pick the "right or best subset of tests" that will unearth maximum number of defects.

Efficiency of test logic can be measured in some of the following ways:

- Average code coverage per test case – (the other side of this is sharpness/ focus of the test case)
- Average state coverage per test case (for state based systems)
- Test thoroughness metric – fault exposure capability of a test case (mutation tests measure this)
- Minimum number of tests required for a given code coverage
- Minimum number of tests required for a given fault exposure

To do non-exhaustive but yet effective testing, techniques like **orthogonal arrays (Taguchi methods)** may be used. This is a technique for picking the most optimal subset (of a given size) of tests by defining the entire input space of the system.

DMADV techniques may also be used for improving test case design. A judicial mix operational profile testing & non-operational profile testing can unearth more defects than any one technique in isolation.

Selection of non-operational profile testing may be guided by some of the following inputs:

- Analysis of metrics gathered during previous releases → categorization of buggy modules/features/business-processes → higher test representation for these
- Analysis of metrics gathered during the earlier phases (design/coding) of the current release → identification of possibly buggy modules/features/business-processes because of poor phase containment → higher test representation for these
- FMEA (failure mode effect analysis) if done for the system → RPN (risk priority number) of modules/features/business-processes → higher representation for those with high RPN

Efficiency of test execution

Test execution efficiency can be improved by automation, usage of tools, shorter test set up times, re-ordering of tests so as to minimize test set up times, better utilization of testing resources (both human & infrastructure), better test process (easier & user friendly test management, result capturing, defect tracking etc..)

TQSS methods may be used for improving efficiency of test execution.

Test Management (process) Tools

Usage or not of a test management tool depends on the size and duration of the testing project and whether the organization already has a test management system or not.

For small projects of short duration commercial test management systems may not be necessary except when organization has one & insists on its usage. However, all records of the testing project should be maintained in an organized manner. Uniform Templates, guidelines, check lists, metrics gathering forms & procedures across the project can be of immense help.

For big projects of long duration, commercial test management systems are recommended. Ex: Mercury International's Test Director or Rational's Test Manager. It could also be the organization's proprietary tool.

The test management tool shall support the following features :

- Test team definition
- Testing work allocation
- Test planning
- Configuration controlled repository of test suites/plans/scripts
- Test Scheduling for automatic execution
- Test result capture
- Test report generation
- Defect tracking
- Multi-site working

Test Automation

Test automation – whether or not, whether only for regression or even the first time, is an ROI based decision. Once decision is made to go for test automation, selecting from the possible options is also important, some of which could be:

- Use of Shell scripts
- Use of TCL
- Use of C/C++
- Use of test harnesses
- Use of TTCN
- Use of Jscript
- Use of record-replay tools
- Use of commercial tools from Rational, Parasoft, Mercury International etc.

Test Suspension & Resumption Criterion

If the system under test is too buggy to allow further testing, test activity will need to be suspended. For each module/ functionality/ release/ test activity, specific test suspension & resumption criteria should defined in this stage.

Metrics that need to be Gathered

The various metrics that need to be collected/ estimated during the various stages & types of testing are defined here. These include product metrics & process metrics.

Process metrics will be fed back to refine the execution test profiles. Product metrics will be fed back to refine test budget distribution, test case design, areas of focus etc.

Process metrics (example list)

- Testing Schedule overrun
- Testing effort overrun
- Phase containment for the various stages of testing
- Number of test cases (manual/ automatic) designed per day/per engineer (productivity)
- Number of test cases (manual/ automatic) executed per day/ per engineer (productivity)
- Occupancy of test bed (productivity)
- Requirement traceability
- Bug fix turn over time

Product metrics (example list)

- Total number of defects detected & categorised by severity/ module/ feature/ requirement/ business process per code size
- Performance metrics (ex: response time for a given transaction)
- Load / stress test metrics (ex: max number of parallel transaction of a given type or mix)
- Estimated failure rate of the delivered software
- Estimate of residual defects left behind in the product
- Estimated reliability growth of the software

Release Test Profile of a S/W Project

Analysis of Test Results – Testing Project Post-Mortem

After results of individual tests are gathered over a length of time, these test results should be subjected to higher level analysis. This analysis should be statistically rigorous & should highlight trends/patterns over long periods of observation. Some things that can be done on a release to release basis are:

- Generation of higher level reports than PASS/FAIL
- Categorization of detected errors/defects on the following basis
- Severity
- Requirement tagged with
- Functionality/ feature affected
- S/w module or component involved
- Business process impacted
- Originating SDLC phase
- Quantitative Determination of a measure of phase containment at various stages of testing - number/ % of errors or defects captured at each stage.
- Trend analysis of test results over a release/ over multiple releases
- Linking trends to SDLC processes, tools used etc...

Determination of Achieved Test Goals & Quality Goals

Quantitative measurement of residual defect density, failure rate, MTBF, RPN, availability etc... , at release, can give a definitive picture of testing effectiveness. This data can be used for comparison against competition & can help in arriving at benchmarks.

Earned Value Analysis

Actual cost of test performed and deviation from estimated cost should be formally determined at product release. This data should be should be used as the input for the next release/or similar project initiation.

This is essential for future testing project cost planning & control.

Conclusion

Thus, defining these three test profiles for a software project can help improve the maturity of software test management.

Note: Appendix on test coverage definitions on the next page.

Appendix – Test Coverage Definitions

Definition of formal & quantitative measure of test coverage is dependent on the type of testing. FORMAL methods (mathematical) or tools which incorporate these will be needed for measuring test coverage.

Some examples of test coverage metrics are given below:

White box Testing:

- Block coverage
- Branch coverage
- Value-range coverage
- Call & return
- % of code exercised or executed
- exception coverage

Block Box Testing:

- Function point coverage
- Feature coverage
- Message coverage
- Unexpected events coverage
- Error code coverage
- Crossing/clashing events/ scenario coverage
- Timer coverage

OO testing:

- Class coverage
- Instance coverage
- Method coverage
- Inheritance coverage

Protocol testing:

- Message coverage
- PDU value range coverage
- State coverage
- Error handling coverage
- Message cross over/ event clash coverage
- Timer coverage

SDL based testing:

- Number of states visited
- Number of transition paths traversed

About the Author

The author has been with WIPRO for 6 years now.For the last two years she has been working in the interops division. Lakshmi is an Electronics H/W & F/W engineer. In WIPRO, she has worked with AGCS & Nortel accounts as a s/w developer before movingto Interops in June 2001.

Since 2001, she has been involved in the LIBRA project & TTCN related activities
In 2001-2002, Lakshmi was involved in a DMADV project, which involved a detailed study of "Reliability Engineering". Some of the thoughts expressed in this paper originate from this study.

In her current role,she is responsible for the "Nokia - Symbian Practice group in Bangalore" & the "TTCN center of excellence" of Interops.

About Wipro Technologies

Wipro is the first PCMM Level 5 and SEI CMMi Level 5 certified IT Services Company globally. Wipro provides comprehensive IT solutions and services (including systems integration, IS outsourcing, package implementation, software application development and maintenance) and Research & Development services (hardware and software design, development and implementation) to corporations globally.

Wipro's unique value proposition is further delivered through our pioneering Offshore Outsourcing Model and stringent Quality Processes of SEI and Six Sigma.

Wipro in Telecom

Wipro Technologies, a leading global provider of IT solutions and services offers a range of services to both incumbents and new age telecom companies. Our target segments include Wireless, Wire line, ISP, Internet Data Center, ASP, and Cable & Satellite. Our key service offerings include Outsourcing Services, Systems Integration, Application Integration Services, Network Management, Integration and Infrastructure Services, Infrastructure Support Services, Custom Development and Maintenance Services.

For further information visit us at http://www.wipro.com/telecom

For more whitepapers logon to: http://www.wipro.com/insights

Worldwide HQ
Wipro Technologies,
Sarjapur Road,
Bangalore-560 035,
India.
Tel: +91-80-844 0011.

U.S.A.
Wipro Technologies
1300, Crittenden Lane,
Mountain View, CA 94043.
Tel: (650) 316 3555.

U.K.
Wipro Technologies
137 Euston Road,
London, NW1 2 AA.
Tel: +44 (20) 7387 0606.

France
Wipro Technologies
91 Rue Du Faubourg,
Saint Honoré, 75008 Paris.
Tel: + 33 (01) 4017 0809.

Germany
Wipro Technologies
Am Wehr 5,
Oberliederbach,
Frankfurt 65835.
Tel: +49 (69) 3005 9408.

Japan
Wipro Technologies
911A, Landmark Tower,
2-1-1 Minatomirai 2-chome,
Nishi-ku, Yokohama 220 8109.
Tel: +81 (04) 5650 3950.

U.A.E.
Wipro Limited
Office No. 124,
Building 1, First Floor,
Dubai Internet City,
P.O. Box 500119, Dubai.
Tel: +97 (14) 3913480.

www.wipro.com
eMail: info@wipro.com